

User name:
Volodymyr Matiievskiy

Check date:
08.01.2023 18:48:42 EET

Report date:
08.01.2023 18:55:23 EET

Check ID:
1013374997

Check type:
Doc vs Internet + Library

User ID:
100010994

File name: **Олейникова М.С**

Page count: **55** Word count: **10772** Character count: **81819** File size: **1.22 MB** File ID: **1013147472**

36.9% Matches

Highest match: **24.8%** with Internet source (<http://ite.kspu.edu/index.php/ite/article/download/434/448>)

36.9% Internet sources 36

Page 57

0.09% Library sources 5

Page 57

2.03% Quotes

Quotes 11

Page 58

Exclusion of references is off

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.

Replaced characters 14

ВСТУП

Сучасна педагогічна наука постійно розвивається й прогресує. Змінюються погляди на педагогічний процес, методи і засоби навчання та виховання стають більш гуманними та ефективними. Визначальною особливістю змін, які відбуваються в освіті України, є перехід до спрямованого формування у суб'єктів навчання здатності творчо діяти, застосовувати знання і досвід на практиці. Зміна цілей навчання визначає необхідність коригування змісту і методик навчання, спрямованого на забезпечення формування світогляду, ціннісних орієнтацій, умінь самостійно вчитися, критично мислити, користуватись сучасними засобами інформаційно-комунікаційних технологій, зокрема персональним комп'ютером; здатності до самопізнання і самореалізації особистості у різних видах творчої діяльності.

Стратегічні орієнтири інформатизації освіти в Україні визначені в Законі України “Про освіту”, Державній національній програмі “Освіта. Україна XXI століття”, Державній програмі „Інформаційні та комунікаційні технології в освіті і науці” на 2006-2010 рр., де наголошується, що розвиток інформаційного суспільства в Україні та впровадження новітніх інформаційно-комунікаційних технологій (ІКТ) в усіх сферах суспільного життя, державного управління, в закладах освіти – один із пріоритетних напрямів державної політики.

Вирішення питань інформатизації навчання різних предметів, зокрема алгоритмізації та програмування, пов'язано з визначенням теоретичних засад організації процесу навчання із застосуванням ІКТ у психолого-педагогічному, дидактичному й методичному аспектах. Насамперед необхідними є наукове обґрунтування, розробка та апробація відповідних засобів навчання алгоритмізації та програмування і способів ефективного поєднання ІКТ із різними методами й формами організації навчання.

Потребують вивчення також методичні засади процесу організації навчальної діяльності студентів на заняттях з алгоритмізації та

програмування із використанням ІКТ у контексті формування в них алгоритмічної компетентності.

Особливого значення при навчанні алгоритмізації та програмування набуває принцип наочності. За рахунок продуманого застосування засобів наочності можна посилити емоційний вплив на студентів, підвищити рівень доступності матеріалу, що вивчається, прискорити активізацію розумової діяльності студентів.

Якщо для удосконалення змісту освіти найважливішими задачами є її оптимізація, систематизація та проблематизація, для удосконалення методики навчання необхідна його індивідуалізація, диференціація та активізація, то найважливішим напрямом удосконалення дидактичних засобів визнається візуалізація. ЮНЕСКО в 2003 році об'явила про пріоритет візуальної подачі матеріалу в освіті.

Наочність у навчанні – один з основних принципів дидактики, відповідно до якого навчання будується на конкретних образах, що безпосередньо сприймаються учнями. Вперше цей принцип обґрунтовано у XVII ст. чеським педагогом Я.А.Коменським, а пізніше Дж.Локком, Ж.Ж.Руссо, Й.Г.Песталоцці, К.Д.Ушинським. Сучасні дидакти розглядають наочність як джерело знань, на основі якого формуються чуттєві уявлення й поняття, як ілюстрацію до положень, що вивчаються, опору для абстрактного мислення. Засоби наочності застосовуються перед вивченням нового матеріалу, в процесі засвоєння понять, повторення й перевірки знань тощо [23].

Як зазначають дидакти XX та XXI ст., зокрема С.П.Баранов, В.І.Бондар, Н.П.Волкова, В.І.Свдокімов, Л.В.Занков, А.І.Зільберштейн, А.М.Маслов, Н.О.Менчинська, М.Г.Моро, А.Розуменко, О.Я.Савченко, Л.М.Скаткін, І.Ф.Харламов, В.В.Ягупов та інші, принцип наочності – це найбільш відомий принцип навчання, який використовується з найдавніших часів. Питання розвитку та адаптації «золотого правила» дидактики до сучасних умов з точки зору використання моделей, процесу моделювання у навчанні

досліджувалося в другій половині ХХ ст. В.Г.Болтянським, Д.Б.Ельконіним, Л.М.Фрідманом та ін. Дослідниками проаналізовано окремі теоретичні та методичні аспекти цієї теми. Особливості реалізації принципу наочності у навчальному процесі в старших класах та у ВНЗ досліджено у роботах Г.Ващенко, С.І.Архангельського. Питання застосування комп'ютерних моделей у ВНЗ знайшли відображення у працях Н.В.Апатової, Т.А.Бороненко, Ю.А.Вороніна, Л.В.Горчакова, І.В.Роберт, І.А.Цвелої та ін. Досить цікавими є роботи Ю.Р.Валькмана, Н.Н.Манько, що стосуються когнітивної візуалізації дидактичних об'єктів для активізації навчальної діяльності.

Використання методу демонстраційних прикладів при навчанні програмуванню базується на концепції відомого методиста в навчанні програмуванню Н.Вірта «Програмування – це мистецтво конструювання. Як можна навчити конструкторській, винахідницькій діяльності. Існує такий метод виділити найпростіші будівельні блоки із багатьох вже існуючих програм і дати їх систематичний опис. Але програмування являє собою велику і різнопланову діяльність, яка часто потребує складної розумової праці.

Помилково вважати, що її можна звести лише до використання готових рецептів. За метод навчання нам лишається обрати ретельний добір і розгляд характерних прикладів. Зрозуміло, не слід вважати, що вивчення прикладів усім однаково корисно. При цьому підході багато залежить від кмітливості й інтуїції учня».

Проблемам ролі візуалізації в підвищенні мотивації та активізації навчальної діяльності з алгоритмізації та програмування присвячені роботи М. С. Львова, Н. В. Морзе, О. В. Співаковського та ін.

При вивченні алгоритмів обробки інформації, які представлені різними структурами даних [14,16], важливу роль відіграють візуалізатори алгоритмів, що дозволяють в наочній формі динамічно відображати деталі їхньої роботи.

Це відкриває можливість використання новітніх технологій при вивченні програмування [13].

Візуалізатор - це програма, в процесі роботи якої на екрані комп'ютера динамічно демонструється застосування алгоритму до вибраного набору даних. Візуалізатори дозволяють вивчати роботу алгоритмів в покроковому режимі, аналогічному режиму трасування програм.

Для деяких алгоритмів динамічний варіант демонстрації його роботи є більш природним, ніж набір статичних ілюстрацій. Для алгоритмів сортування візуалізація дозволяє наочно продемонструвати як загальний підхід, так і відмінність у механізмах їх дії.

В результаті візуальної підтримки алгоритмічної підготовки студентів краще засвоюється навчальний матеріал, його усвідомлення відбувається діяльнісним шляхом, і як наслідок, студенти знаходять можливості його широкого практичного застосування.

Використання візуалізаторів при формуванні алгоритмічних компетенцій дозволяє розвивати пізнавальні можливості студентів, вміння самостійно аналізувати та інтерпретувати результати, спонукає до дослідницької діяльності. Таким чином, у студентів формуються якісно нові професійно значимі вміння та навички, реалізується підготовка майбутнього спеціаліста для успішної професійної діяльності.

Таким чином розробка Web-модуля навчальні системи для візуалізації алгоритмів сортування є актуальною темою.

Об'єкт дослідження – візуалізатори алгоритмів.

Предмет дослідження – навчальні системи для візуалізації алгоритмів сортування.

Мета роботи - аналіз візуалізації алгоритмів сортування та розробка Web-модуля навчальної системи для візуалізації алгоритмів сортування.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Розглянути застосування візуалізаторів в навчальному процесі.

2. Проаналізувати та порівняти алгоритми сортування.
3. Визначити зміст, структуру, функції і дидактичні можливості Web-модуля навчальної системи для реалізації візуалізації алгоритмів сортування.
4. Вивчити програмне забезпечення, яке використовується для створення додатку.
5. Розробити Web-модуль навчальної системи для візуалізації алгоритмів сортування.

Методи дослідження: методи теорії алгоритмів та мов програмування, методи комп'ютерної графіки.

Практичною цінністю роботи є розроблений Web-модуль навчальної системи для візуалізації алгоритмів сортування, який можливо використовувати в навчальному процесі.

В першому розділі розглянуто опис поточного стану в області візуалізаторів і систем візуалізації. Зокрема, розглянуто застосування візуалізаторів в навчальному процесі і виникаючі при цьому вимоги до візуалізаторів. У цьому розділі також наводиться аналіз візуалізаторів алгоритмів з точки зору висунутих вимог.

В другому розділі розглядається оцінка складеності алгоритмів сортування. Проводиться аналіз алгоритмів сортування.

У третьому розділі перебуває розроблювальна частина роботи. У ньому обґрунтовується вибір технології та середовища розробки, розглядається розробка Web-модуля навчальної системи для візуалізації алгоритмів сортування. Так само в цьому розділі містяться методичні вказівки по роботі з додатком.

РОЗДІЛ 1. ВІЗУАЛІЗАТОРИ АЛГОРИТМІВ

1.1. Застосування візуалізаторів в навчальному процесі

1.1.1. Реалізація дидактичного принципу наочності в алгоритмічній підготовці студентів

Необхідність використання наочного матеріалу в процесі навчання вперше теоретично обґрунтував у XVII ст. чеський педагог Я.А.Коменський. Він визначив, що основою навчання має бути чуттєве пізнання. «Для того, щоб все сприймалося легше, потрібно, наскільки лише це можливо, залучати до сприймання зовнішні чуття» [15].

Видатний педагог сформулював «золоте правило»: «Нехай буде для учнів золотим правилом все, що тільки можна давати для сприйняття чуттями, а саме: видиме – для сприйняття зором, що чуємо – слухом, запах – нюхом, що підлягає смаку – смаком, доступне дотику – шляхом дотику. Якщо будь-які предмети одразу можна сприймати декількома чуттями, нехай вони відразу охоплюються декількома чуттями. От же, чим більше знання базується на відчутті, тим воно достовірніше» [15]. Величезною заслугою автора «Великої дидактики» є розкриття необхідності використання моделей, як прототипу оригіналу. Ці моделі наділені ознаками чуттєвого сприймання і споглядання та використовуються для показу «пізнання предметів».

У XVIII – поч. XIX ст. видатний швейцарський педагог Й.Г.Песталоцці обґрунтував теорію наочного навчання. Він вважав наочність абсолютною основою будь-якого пізнання і наголошував, що чуттєве сприйняття є єдиним фундаментом будь-якого людського пізнання. Наочне навчання є «лише простим продовженням того, що було закладено природою в якості інстинкту і навіть у тих же природних проявах» [21].

Наочне навчання, за Й.Г.Песталоцці, досить близьке за змістом до «золотого правила»: «Чим більшою кількістю чуттів ти пізнаєш суть явища чи будь-якого предмета, тим правильнішими будуть твої знання про нього» [21]. Незважаючи на єдність і спільність розуміння основи наочного навчання цими двома педагогами, Й.Г.Песталоцці, на відміну від А.Я.Коменського:

а) розуміє наочне навчання не лише як отримання знань на основі спостереження, власного чуттєвого досвіду, а бачить у ньому важливий фактор розвитку логічного мислення;

б) дає психологічне пояснення наочного навчання, зазначаючи, що воно ґрунтується на властивості людського розуму узагальнювати враження, отримані від природи через органи чуттів, у деяку єдність – поняття, й поступово доводити їх до чіткого розуміння [20].

У розумінні видатного педагога XIX ст. К.Д.Ушинського наочне навчання – це таке навчання, яке ґрунтується не на абстрактних уявленнях і словах, а на конкретних образах, безпосередньо сприйнятих дитиною або під керівництвом наставника, або завдяки самостійним спостереженням. Разом з педагогічним аналізом наочного навчання К.Д.Ушинський виконує його психологічний аналіз, автор пише, що чим більше органів наших чуттів бере участь у сприйманні будь-якого враження чи групи вражень, тим міцніше лягають ці враження в нашу механічну, нервову пам'ять і надійно зберігаються нею та легше потім пригадуються. К.Д.Ушинський розуміє наочність як дидактичний принцип, який пронизує як зміст, так і окремі методи та прийоми навчання і взаємопов'язаний з усіма іншими дидактичними принципами, які є «необхідними умовами навчання».

У «Педагогічному словнику» зазначалося, що наочність у навчанні – дидактичний принцип, згідно з яким навчання будується на конкретних образах, безпосередньо сприйнятих учнями. Таке розуміння суті наочного навчання було панівним, домінуючим у педагогічній науці XX ст. Так, А.І.Зільберштейн відзначає, що значення принципу наочності визначається теорією, в основі якої «лежить визнання зовнішнього світу і відбиття його в людській голові. Усі знання люди набувають із зовнішнього світу через відчуття і сприймання. Без них не може розвиватися й мислення».

70 – 80-ті роки відзначаються роботами В.Г.Болтянського, В.В.Давидова, Д.Б.Ельконіна, Л.М.Фрідмана. Як стверджують літературні

джерела, питання моделювання в процесі навчання першим порушив Д.Б.Ельконін.

Психологи XX століття багато уваги приділяли сприйняттю учнем та студентом предметів та явищ оточуючого світу. В результаті більшість з них прийшли до висновку, що «наочність не ізолює сприйняття і представлення від цілісної аналітико-синтетичної розумової діяльності».

XXI ст. вносить свої корективи в розуміння та використання наочності в дидактиці, зумовлюючи появу як нових термінів – «моделювання», «нова наочність» та інші, так і розуміння їх специфіки, необхідності та призначення на сучасному етапі.

У сучасній педагогіці трактування принципу наочності співпадає з розумінням наочного навчання Я.А.Коменським, Й.Г.Песталоцці, К.Д.Ушинським. Так, у підручнику з педагогіки Волкової Н.П. (2001 р.) зазначається я: «Принцип наочності передбачає навчання на основі живого сприймання конкретних предметів і явищ дійсності або їх зображень».

Наочне навчання передбачає, що у процесі пізнання повинні застосовуватися різні відчуття, в тому числі шляхом зорового сприймання. Сприйняті речі, вважає Г.Вашенко, залишають у нашій свідомості певні образи, уявлення. На їх основі й розвиваються вищі форми мислення. Особливо це стосується дитинства, коли один лише вербальний спосіб подачі інформації сприяє формуванню так званого вербального типу мислення, що характеризується поверхневістю і неповною відповідністю між словом і думкою. Зрештою, і цим принципом, на думку Г.Вашенка, зловживати не варто, особливо в роботі з учнями старших класів, де акцент переноситься на рівень абстракції.

У технічних навчальних закладах наочність розглядання навчального матеріалу доповнюється використанням технічних засобів, які дозволяють розширити межі можливостей чутливих органів людини. С.І. Архангельський визначає значення технічних засобів таким чином: «Технічні засоби навчання розширюють змістовний бік наочності навчання, дозволяють передавати

інформацію в більш активній формі сприйняття, вони накладають свій відбиток на мислену діяльність студентів, їх емоційний стан, змінюють їх психічне навантаження».

При вивченні певного предмету наочність навчання поєднує в собі дві сторони пізнання – чутливу та мислену і сприяє розкриттю зовнішніх ознак та властивостей цього предмету. На думку С.І. Архангельського, однією з важливіших функцій наочності є створення уявлень, які стають основою понять. Він розглядає наочність як «перехід у навчанні від конкретного до абстрактного, від суцього до мислі, від ознак і уявлень до понять і визначень». Головною задачею наочної в навчанні стає забезпечення зв'язку між ознаками, що спостерігаються і уявленнями, що створюються з свідомим і глибоким розумінням сутності предмета, який вивчається студентом. С.І. Архангельський поділяє наочність у навчанні на два типи: «безпосередня наочність, що заснована на спостереженнях дійсності та опосередкована наочність, яка визначає явище, подію, предмет вивчення у певній наглядній формі, яка відображає його сутність, зв'язки і відношення». Таким чином, наочність у навчанні забезпечує початкове розкриття зовнішніх ознак і властивостей об'єкту, що вивчається. Вона стимулює пізнавальну діяльність студента та активізує механізм сприйняття навчальної інформації.

В.М. Вергасов висловлює думку, що «наочність бере участь у формуванні первинної нейронної моделі образу, поняття, явища на етапі сприйняття. Тому наочність повинна якомога більше підкріплювати цей етап переходу інформації із зовнішнього середовища у пам'ять інтелекту».

Навчальні функції наочності були досліджені у традиційній дидактиці такими педагогами дослідниками, як Л.В. Занков, Ф.І. Менчинська, М.І. Махмутов, В.А. В'ялих та ін.

Л. В. Занков виділяє три основні функції, де наочність виступає як:

- джерело інформації;
- засіб ілюстрації інформації;

- опора для усвідомлення зв'язків між явищами, предметами та поняттями.

М.І.Махмутов додає ще одну функцію, в якій наочність розглядається як засіб формування проблемних ситуацій, що в свою чергу стимулює розвиток творчих навиків при самостійній діяльності майбутнього фахівця. В.А.В'ялих приписує наочності функцію активізації пізнавальної діяльності студента, розглядає її як основу абстрактного мислення.

Основою сучасної теорії наочного навчання є використання надбань минулого з врахуванням вимог сьогодення. Тому, традиційного розуміння принципу наочності у процесі пізнання вже недостатньо. Це питання потребує розвитку та вдосконалення. На нашу думку, саме застосування моделювання у процесі навчання є наступним рівнем використання наочного матеріалу й новим етапом в історії його розвитку.

Сучасний розвиток інформаційних технологій та комп'ютерної техніки дає можливість використовувати наочність зовсім на іншому рівні, збільшити її інформаційну та пізнавальну складову.

Комп'ютеризація освітнього процесу відкриває нові шляхи в розвитку мислення, надаючи нові можливості для активного навчання. Оскільки наочно-образні компоненти мислення відіграють виключно важливу роль у житті людини, то використання їх у навчальному процесі виявляється досить ефективним. Комп'ютерна графіка може використовуватися на всіх етапах процесу навчання: при поясненні нового матеріалу, закріпленні, повторенні, контролі. Таким чином стає необхідним перейти від погляду на наочність як одного з допоміжних засобів навчання алгоритмізації до повноцінного використання візуального мислення в процесі алгоритмічної підготовки студентів.

За визначенням В.П.Зінченко: «Візуальне мислення – це людська діяльність, продуктом якої є породження нових образів, створення нових візуальних форм, що несуть певне смислове навантаження і що роблять значення видимим».

Зміст принципу наочності в сучасному його розумінні визначає логіку пізнання від чуттєво-наочного до абстрактно-логічного, від наочності чуттєво-конкретної (об'єкти в натурі, малюнки, макети та ін.) до наочності абстрактної і символічної (схеми, таблиці, діаграми, графіки). Наочність пов'язана з роботою органів чуття (аналізаторів), зорових, слухових, тактильних тощо. Однак з'ясувалося, що принцип наочності виявився місткішим.

Тепер ідеться про роль наочності як засобу переходу від чуттєвого матеріалу до його абстрактного тлумачення і від абстрактного до глибшого пізнання чуттєвого. Чуттєвий матеріал – зміст наочності, таких її видів, як природна наочність (натуральні предмети й об'єкти), образотворча наочність (малюнки, фотографії та ін.), реальні моделі. На етапі переходу до абстрактних понять необхідні інші засоби наочності – схеми, таблиці, графіки, символи. Цей вид наочності – абстрактно-символічний. Він допомагає досягнути сутність і динаміку явищ і процесів, що їх досліджують.

Використання наочності потрібно підпорядковувати конкретній меті, розвитку самостійності й активності студентів з урахуванням їх вікових особливостей. Вона має бути змістовною, естетично оформленою, відповідати психологічним законам сприймання.

Тому необхідно дотримуватися правил реалізації принципу наочності:

1. Запам'ятовування предметів у натурі, на картинах або моделях відбувається краще і швидше, ніж запам'ятовування поданого словесно, усно або письмово.
2. Дитина мислить формами, фарбами, звуками, образами взагалі: звідси доцільність наочного навчання, яке будується на конкретних образах.
3. Золоте правило: що тільки можна – діти мають сприймати відчуттями (зором, слухом, нюхом тощо). Це стосується передусім процесу початкового навчання.
4. Наочність – не мета, а лише засіб досягнення поставленої мети.

5. Поняття доходять до свідомості учнів легше, коли вони підкріплені конкретними фактами, прикладами та образами. Для розкриття їх необхідно застосовувати всі види наочності.
6. Слід використовувати наочність як самостійне джерело інформації для створення проблемних ситуацій! Сучасна наочність дає змогу організувати колективну пошукову та дослідницьку роботу учнів.
7. Спостереження учнів повинні бути систематизованими і перебувати у співвідношенні причини й наслідку не залежно від часу їх набуття.
8. Застосовуючи наочні засоби, необхідно розглядати їх з учнями спочатку загалом, потім – головне й другорядне, наостанку – знову загалом.
9. Надмірна кількість наочних посібників розсіює увагу учнів і заважає досягнути головного.
10. Використовуючи наочність, необхідно актуалізувати чуттєвий досвід учнів: уявлення, які вже склалися в дітей, конкретизувати та ілюструвати ті поняття, які формуються.
11. Необхідно намагатися виготовляти наочні посібники разом зі своїми учнями.
12. Необхідно старанно готувати наочність до занять.
13. Необхідно науково обґрунтовано застосовувати сучасні засоби наочності: полієкранну проєкцію, навчальне телебачення, відеозапис, кодослайди, комп'ютери, проєктори, тощо; досконало володіти технічними засобами, методикою їх використання.
14. В умовах кабінетної системи навчання можливості застосування наочності кращі, це вимагає ретельного планування й дозування наочності.
- 15.3 віком учнів предметна наочність повинна дедалі більше поступатися місцем символічній. Особливу увагу вчитель має

приділяти адекватно сті розуміння суті явищ і їх наочного подання.

16.За надмірного захоплення наочністю створюються штучні перепони на шляху до глибокого оволодіння знаннями: вона стає гальмом розвитку абстрактного мислення, розуміння суті загальних закономірностей.

Необхідним компонентом правильно побудованого процесу навчання є дидактичні засоби. Як підкреслює В. Оконь, не дивлячись на те, що дидактичні засоби не здійснюють вирішального впливу на кінцеві результати навчально-виховної роботи, тим не менш, збагачуючи методи навчання, вони сприяють росту їх ефективності. Правильно підібрані й уміло включені в систему використовуваних викладачем методів та організаційних форм навчання, дидактичні засоби полегшують реалізацію принципу наочності. Завдяки цьому вони не тільки покращують умови безпосереднього пізнання дійсності учнями та студентами, але і дають матеріал у формі вражень і спостережень, на який спираються опосередковане пізнання, розумова діяльність, а також різні види практичної діяльності.

Одним із ефективних методів, що сприяють розвитку алгоритмічної та технологічної компетентності, на нашу думку, є застосування комп'ютерних моделей різних інформаційних процесів та об'єктів у процесі навчання алгоритмізації.

Дослідники приводять наступні програмні засоби, що використовуються в процесі алгоритмічної підготовки: виконавці алгоритмів; комп'ютерні навчальні середовища або мікросвіти; електронні тренажери; імітатори; моделі інформаційних процесів та обчислювальних систем та ін. Важливого значення серед них набувають візуалізатори алгоритмів.

Нові властивості візуальних засобів, які не вивчалися в педагогічній науці в зв'язку з поняттям «наочність», а стали відомі завдяки дослідженням в області візуалізації дидактичних об'єктів, повинні сприяти перетворенню

базових елементів освітнього процесу (вивчаємий дидактичний об'єкт, навчальна діяльність і суб'єкт навчання).

Виходячи з того, що моделювання є «активною» і найбільш продуктивною формою діяльності, була визначена необхідна педагогічна умова позитивної активізації навчальної діяльності – формування моделюючого середовища, що включає технологію, за собою і способи візуалізації дидактичних об'єктів. Моделювання, що розглядається як спосіб довільного формування когнітивно-візуальних образів вивчаемого об'єкта і оперування їх властивостями для раціоналізації пізнання, застосування і перетворення дійсності, дозволяє створювати семантичний простір дослідження, забезпечує можливість експериментування на моделі, включаючи логічне узагальнення, рефлексію та інші форми розумової діяльності [18].

Важливі проблеми виявлення потенціалу когнітивної візуалізації в алгоритмічній підготовці студентів до сліджуються в НДІ ІТ Херсонського державного університету, де під керівництвом М.С. Львова та О.В. Співаковського розроблено програмно-методичний комплекс (ПМК) «Відеоінтерпретатор алгоритмів пошуку та сортування» [17].

1.1.2. Візуалізатори в навчальному процесі

Візуалізатори алгоритмів широко застосовуються для навчання інформатики [11, 22]. Дослідження [30] показали, що застосування візуалізаторів допомагає учням глибше і швидше зрозуміти з'ясовний матеріал. При цьому так само виявлена велика зацікавленість учнів у лекціях із застосуванням візуалізаторів, ніж без таких.

Процес навчання при використанні для вивчення алгоритмів книг відбувається одним із таких способів:

1. Статичне сприйняття тексту з динамічною прокруткою в голові.
2. Реалізація алгоритму на вибраній мові програмування, або копіювання алгоритму з книги з тим, щоб динамічно, крок за кроком, відстежити дію алгоритму на тестових прикладах.

Перший спосіб є досить складним для більшості учнів, оскільки вимагає програмістської уяви. Він доступний лише досвідченим програмістам, а не учням старших класів школи або молодших курсів інституту, які тільки приступають до вивчення алгоритмів. Другий спосіб є більш прямолінійним і досить тупим, однак акцентує увагу не на суті алгоритму, а на його програмній реалізації і тонкощах використовуваної мови програмування.

Таким чином, традиційне статичне викладання матеріалу по алгоритмізації та програмування є неефективним з точки зору вивчення алгоритмів. Крім того, якщо розглянути наведені способи навчання з точки зору використання на лекціях з алгоритмізації та програмування, то очевидно, що вони також не ефективні. Якщо перший спосіб вимагає певних зусиль, то другий і зовсім неможливо реалізовувати на лекційних заняттях.

Подальші дослідження в галузі викладання алгоритмізації та програмування призвели до виникнення в середині вісімдесятих років [22] нового підходу до викладання - з'явилися візуалізатори алгоритмів дискретної математики.

Візуалізатор - це програма, яка ілюструє виконання алгоритма при певних вхідних даних. Прикладами візуалізаторів можуть служити, наприклад, програма, що займається побудовою графіка функції, або програма, візуально моделюючи який-небудь фізичний процес. Стосовно до дискретної математики й програмування, візуалізатори зазвичай моделюють деякі алгоритми, даючи можливість навчаючому за допомогою інтуїтивно зрозумілого інтерфейсу проходити алгоритм крок за кроком від початку до кінця, а при необхідності, і навпаки.

Перерахуємо відмітні характеристики візуалізаторів.

1. Простота використання, обумовлена зрозумілістю інтерфейсу. Тому для роботи з візуалізатором зазвичай не потрібна спеціальна підготовка.
2. Чіткість і простота представлення візуалізованого процесу.

15

3. Компактність візуалізаторів. Це при необхідності спрощує передачу візуалізаторів в мережі Інтернет, що особливо важливо при дистанційному навчанні.

Візуалізатори в розглянутій області вирішують наступні завдання, що виникають в процесі навчання.

1. Графічне та текстове роз'яснення дій алгоритму на конкретних наборах вхідних даних. При цьому розуміння алгоритму не потрібно, оскільки саме візуалізатор повинен пояснити дію алгоритму.
2. Надання користувачеві інструменту, що реалізує даний алгоритм. У результаті учень звільняється від необхідності виконувати кроки алгоритму, так як їх автоматично виконує візуалізатор.

Візуалізатор виконує зазвичай такі функції :

1. Покрокове виконання алгоритму.
2. Перегляд дії алгоритму при різних наборах вхідних даних, у тому числі і введених користувачем.
3. Перегляд дії алгоритму в динаміці.
4. Перезапуск алгоритму на поточному наборі вхідних даних.

Динамічна візуалізація наочно демонструє таку характеристику алгоритму, як трудомісткість (особливо при пошаговій демонстрації). Для деяких алгоритмів (наприклад, машини Тюринга) динамічний варіант демонстрації взагалі представляється більш натуральним, ніж будь-який набір статичних ілюстрацій.

У завдання візуалізатора алгоритму входить:

1. Відображення вхідних і вихідних даних у наочній формі- уявлення абстрактного поняття дано в відекартинках. Таке уявлення необхідно, як для учнів з початковим рівнем підготовки, так і для більш продвинутих учнів з метою спрощення сприйняття.
2. Відображення внутрішніх службових змінних.

3. Відображення процесу впливу алгоритму на вхідні дані та внутрішні змінні:
- a) Зміна даних.
 - b) Копіювання і переміщення даних.
 - c) Процес прийняття рішень.
 - d) Коментарі до кожної дії алгоритму.
 - e) Відображення роботи алгоритму по кроках.
 - f) Можливість введення даних з метою апробації алгоритму на різних наборах вхідних даних.

1.1.3. Варіанти застосування візуалізатора

При навчанні візуалізатори можуть бути використані декількома різними способами. Опишемо деякі з них.

На початку візуалізатори алгоритмів використовувалися як супроводжуючий матеріал на лекціях. При цьому, викладач заздалегідь готує візуалізаційний матеріал і показує його на лекціях, пояснюючи роботу алгоритму. Візуалізатори так само можуть бути використані для більшого залучення учнів в процес навчання. Одним з таких способів є відображення деякого стану алгоритму і пропозиція учням визначити, яку дію виконує алгоритм. Після отримання відповідей, вони перевіряються шляхом виконання кроків вперед. Візуалізатор так само дозволяє детально розібрати чому було здійснено та обрано саме цю дію, а не другу. У такому режимі візуалізатори так само можуть бути використані для перевірки знань.

Інший спосіб використання візуалізаторів - початкове знайомство з алгоритмом. Учень спочатку працює з візуалізатором, складаючи для себе загальну схему алгоритму. Пізніше викладач дає повний опис алгоритму, після чого учень може повернутися до роботи з візуалізатором.

Ще один спосіб використання візуалізаторів полягає в тому, що учні самі строять візуалізатори, досконально вивчаючи при цьому візуалізуючі алгоритми.

Отримані таким чином візуалізатори можуть бути опубліковані в мережі Internet і використані наведеними вище способами.

Важливою областю використання візуалізаторів є дистанційне і самостійне навчання. У цьому випадку велике значення має можливість завдання користувачем вхідного набору даних.

Таки чином, учень може не запитувати викладача, що буде при обробці деякого набору даних, а ввести його в візуалізатор і подивитися самостійно.

1.1.4. Вимоги до візуалізаторів алгоритмів

Для використання в навчальному процесі візуалізатори алгоритмів повинні задовольняти наступним вимогам:

1. Керованість - в учнів повинна бути можливість задавати власні набори вхідних даних і розглядати роботу алгоритму на них.
2. Інтерактивність - при роботі з візуалізатором повинна бути можливість здійснювати кроки як вперед, так і назад, також повинен бути передбачений режим автоматичної візуалізації.
3. Історія - візуалізатор повинен дозволяти зробити скільки завгодно багато кроків назад, відображаючи стан алгоритму на відповідний момент.
4. Можливість відображення ходу виконання алгоритму.
5. Можливість коментування виконання програми.
6. Простота використання - візуалізатор повинен бути зрозумілим для не підготовленого користувача.
7. Доступність - в учнів повинна бути можливість доступу до візуалізаторів не тільки на заняттях.
8. Платформонезалежність - візуалізатор повинен працювати не залежно як від платформи (IBM-PC сумісні комп'ютери, Apple Macintosh) так і операційної системи (Windows, Unix, Linux).
9. Незалежність від мережі - для роботи візуалізатора не повинен бути потрібен доступ до мережі.

10.Зручність створення візуалізаторів алгоритмів – простота використання системи візуалізаторів з точки зору написання нових візуалізаторів.

11.Керованість, інтерактивність і історія дозволяє більш глибоко вивчити роботу алгоритму, ніж візуалізація на заздалегідь обраних наборах даних.

Простота використання важлива при самостійному навчанні, так як в цьому випадку користувачі дуже рідко читають інструкції по застосуванню. Можливості відображення шагу алгоритму і коментування шагу програми є дуже важливими. Перша - для пояснення простих алгоритмів і введення в інформатику, а друга - для розуміння складних алгоритмів. Доступність, платформонезалежних і незалежність від мережі дозволяє використовувати візуалізатори як на лекціях і практичних заняттях, так і для самостійного навчання.

1.2. Огляд візуалізаторів на прикладі алгоритмів сортування

Зробимо огляд візуалізаторів алгоритмів на прикладі алгоритмів сортування, так як вони є однією з основоположних тем при навчанні алгоритмізації та програмування і входять як важлива складова частина до багатьох алгоритмів.

1.2.1. Підходи до візуалізації алгоритмів сортування

Розглянемо кілька підходів до візуалізації алгоритмів сортування:

1. «Числовий» - сортований масив відображається набором чисел, які містяться в ньому. При порівнянні і обміні елементів вони підсвічуються. У деяких випадках застосовується анімація обміну елементів [26];
2. «Гістограммний» - сортований масив зображується у вигляді гістограми. При цьому обмін елементів візуалізується як фізичне переміщення стовпців, відповідних чисел [27];
3. «Історичний» - весь процес сортування представляється у вигляді одного зображення. При цьому зазвичай значення елементів відображають кольором [26].

На рис. 1.1. наведені приклади зображень, які генерують візуалізатори, що використовують описані підходи.



Рис. 1.1. Підходи до візуалізації алгоритмів сортування «числовий» (а), «гістограмний» (б) та «історичний» (в)

Зазвичай візуалізується наступний «стандартний» набір алгоритмів сортування: обмінна, бульбашкова, простого вибору, простої вставки, пірамідальна, злиттям і швидка.

1.2.2. Огляд візуалізаторів алгоритмів сортувань

У наведеному нижче огляді в заголовку підрозділів наводиться назва університету, в якому виконаний візуалізатор.

Brown University (BALSA)

Система візуалізації BALSA є одним з перших дослідів створення візуалізаторів алгоритмів. Принципи побудови цієї системи описані в роботі [26].

Ця система дозволяє проводити візуалізацію одночасно на основі кількох підходів, наприклад, «гістограмного» і «числового». При цьому відповідні зображення оновлюються синхронно зі зміною даних. Також є можливість використовувати «історичний» підхід, при якому на одному зображенні показується процес сортування в цілому.

Візуалізатори, в основному, демонструють зміни в сортованих даних. При цьому відображення алгоритму та коментарів відсутня, а процес візуалізації коментується викладачем. Сама система дозволяє відображати коментарі і кроки алгоритму, але це практично не використовується.

Візуалізація виконується за допомогою заздалегідь написаних програм на мові сценарії в. Можливість ручного введення даних не передбачена.

Візуалізатори виконуються у вигляді модулів, що вбудовуються в систему Balsa, і не є платформонезалежними. Це також обмежує їх доступність.

State University of New York, College at Brockport

Колекція візуалізаторів сортувань розташована за адресою [32]. Представлений «стандартний» набір візуалізаторів сортувань, створених на загальному ядрі. При цьому для алгоритму пірамідального сортування використано спеціальне рішення.

При візуалізації застосовується «гістограммний» підхід. Візуалізація виконується на заданому при створенні візуалізатора наборі даних. Існує можливість здійснювати кроки назад за алгоритмом, але перевірка, виконана автором, показала, що вона не працює на деяких наборах вхідних даних.

Присутні коментарі до поточних дій, але алгоритми представлені тільки на супровідних сторінках. Візуалізатори виконані у вигляді Java-апплетів, що визначає їх доступність і платформонезалежність. Для візуалізації зв'язку з сервером не потрібно. Таким чином, ці візуалізатори є автономними.

Princeton University

Візуалізатори виконані у вигляді єдиного Java-апплета, доступного за адресою [31]. На додаток до «стандартного» набору візуалізаторів сортування візуалізуються алгоритми побудови опуклих оболонок. Для візуалізації застосовується як «числовий», так і «гістограммний» підходи з можливістю вибору. Вихідний набір даних може бути задано тільки випадково.

При візуалізації алгоритми не відображаються, так як використана візуалізація даних. З тієї ж причини вироблені дії не коментуються. Третій недолік - неможливість здійснювати шаги назад за алгоритмом.

Hope College

Візуалізатори сортування, виконані в єдиній оболонці, доступні за адресою [28]. Для візуалізації використовується «гістограммний» підхід. Введення вхідних даних неможливе. Трасування алгоритму у зворотньому напрямку не передбачено.

При візуалізації відображається код програми і підсвічується поточний оператор, але вироблені дії не коментуються. Візуалізатор виконаний у вигляді Java-аплета і є платформонезалежним і автономним.

University of Joensuu (Jeliot)

У прикладах до системи візуалізації Jeliot [29] наведені тільки візуалізатори бульбашкової та швидкої сортувань.

При виконанні алгоритмів використовується низькорівнева візуалізація, що відображає не тільки виконання операторів, але й розрахунки виразів. При цьому підсвічується поточний оператор. Введення вихідних даних виконується через спеціальні класи вводу-виводу. Трасування алгоритму у зворотному напрямку неможливе. Для перегляду візуалізаторів, побудованих на базі системи Jeliot, потрібна присутність самої системи візуалізації, виконаної в вигляді Java-додатки, що постачаються для різних платформ. Таким чином, система не є в повній мірі платформонезалежною.

СПбДУ ІТМО (Казаков М.А.)

В роботі [12] описаний візуалізатор пірамідального сортування, доступний за адресою [25].



Рис. 1.2. Візуалізатор пірамідального сортування

Даний візуалізатор заснований на автоматном підході, викладеному в роботі [24]. При візуалізації на екрані відображаються стан масиву, піраміда та коментар до виконуваної дії. Введення вихідних даних непередбачено.

Алгоритм трасується тільки в прямому напрямку. Візуалізатор виконаний у вигляді Java-аплета і є платформонезалежним.

НДІ ІТ Херсонський державний університет (М.С.Львов та О.В.Співаковський)

Програмно-методичний комплекс “Відеоінтерпретатор алгоритмів пошуку та сортування” призначений для застосування при вивченні дисципліни "Основи алгоритмізації і програмування" студентами ВНЗ як засіб вивчення алгоритмів, мови програмування, налагодження програм, поліпшення логіки розробки алгоритмів і програм [17].

Клас задач ПМК – різні алгоритм и обробки масивів даних, у тому числі сортування, пошук унікальних елементів (максимуми, мінімуми і т.і.). За його допомогою можна вивчати теми, пов’язані з мовою програмування, допоміжними алгоритмами, рекурсією тощо. Робоча мова програмування ПМК – Паскаль. Головною перевагою ПМК є візуалізація процесу виконання алгоритмів в динаміці, яка сприяє кращому розумінню основних понять алгоритмізації та програмування.

Логічним розвитком цього додатку стало розроблене у 2007-2009 рр. WEB-орієнтоване Інтегроване середовище курсу «Основи алгоритмізації та програмування» для вищих навчальних закладів, створене для застосування в навчальному процесі при вивченні тем, пов’язаних з алгоритмами обробки масивів, задач вибору, пошуку та впорядкування даних.

Інтегроване середовище курсу «Основи алгоритмізації та програмування» складється з наступних модулів: електронний посібник, бібліотека лекцій, бібліотека задач, середовище демонстрації програм (рис. 1.3), система поточного та підсумкового контролю знань, що містить алгоритмічні тести, електронний журнал.



Рис. 1.3. Візуалізація алгоритму сортування вставками у середовищі демонстрації інтегрованого середовища вивчення курсу «Основи алгоритмізації та програмування»

Модуль «Середовище демонстрації» призначений для використання на лекціях, при проведенні практичних і лабораторних робіт для наочної демонстрації виконання алгоритмів та організації обчислювального експерименту з аналізу їх ефективності. Незаперечною перевагою модуля «Середовище демонстрації» є можливість візуалізації, як класичних алгоритмів, що знаходяться у колекції системи, так і алгоритмів, розроблених користувачем.

1.2.3. Аналіз візуалізаторів алгоритмів сортувань

Розглянемо описані візуалізатори з точки зору вимог, викладених у розділі 1.1.4. Порівняльна характеристика розглянутих візуалізаторів приведена в таблиці 1.1. Її стовпці відповідають властивостям, зазначеним в розділі 1.1.3. При цьому використовуються такі позначення:

- «+» - Властивість виконується;
- «-» - Властивість не виконується;
- «±» - властивість виконується частково.

Таблиця 1.1.

Порівняльна характеристика візуалізаторів алгоритмів сортувань

Візуалізатори	1	2	3	4	5	6	7	8	9	10
Brown University	-	+	±	+	±	±	-	-	-	-
SUNY Brockport	-	±	+	±	-	+	-	+	+	+
Princeton University	±	-	+	-	-	-	±	+	+	+
Hope College	-	-	+	-	+	-	+	+	+	+
Jeliot	+	-	-	-	+	-	+	+	±	+
СПбГУ ИТМО (Казаков М.А.)	-	-	+	-	-	+	+	+	+	+
НДІ ІТ ХДУ (М.С.Львов та О.В.Співаковський)	±	+	±	+	+	+	+	+	±	+

З наведеної таблиці випливає, що не один з розглянутих візуалізаторів не задовольняє всім висунутим вимогам. Тому необхідно розробити нову систему візуалізації, для якої всі зазначені властивості виконуються.

1.3. Висновки до розділу

В данному розділі розглянуто застосування візуалізаторів в навчальному процесі, зокрема сформульовано вимоги до візуалізаторів, які застосовуються при навчанні. Далі розглянуті поточний стан загальнодоступних візуалізаторів на прикладі візуалізаторів сортувань і вказані їх недоліки.

РОЗДІЛ 2. АНАЛІЗ АЛГОРИТМІВ СОРТУВАННЯ**2.1. Оцінка складності алгоритмів сортування**

Нехай A — алгоритм розв'язання деякого класу задач, а N — розмірність окремої задачі цього класу. N може бути, наприклад, розмірністю оброблюваного масиву, числом вершин оброблюваного графа тощо. Позначимо функцію, що дає верхню межу максимального числа основних операцій (додавання, множення і т. д.), які повинен виконати алгоритм A , розв'язуючи задачу розмірності N . Говоритимемо, що алгоритм A поліноміальний, якщо зростає не швидше, ніж деякий поліном від N . В іншому разі A — експоненціальний алгоритм.

Виявляється, що між цими класами алгоритмів є істотна різниця: при великих розмірностях задач (які, зазвичай, найцікавіші на практиці), поліноміальні алгоритми можуть бути виконані на сучасних комп'ютерах, тоді як експоненціальні алгоритми для практичних розмірностей задач, як правило, не можуть виконатися повністю. Зазвичай розв'язання задач, що породжують експоненціальні алгоритми, пов'язаний з повним перебором всіх можливих варіантів, і зважаючи на практичну неможливість реалізації такої стратегії, для їх розв'язання розробляються інші підходи.

Наприклад, якщо навіть існує експоненціальний алгоритм для знаходження оптимального розв'язку деякої задачі, то на практиці застосовуються інші, ефективніші поліноміальні алгоритми для знаходження не обов'язково оптимального, а лише прийнятного розв'язку (наближеного до оптимального). Але навіть якщо задача допускає розв'язання за допомогою поліноміального алгоритму, його можна побудувати лише після глибокого вникання в суть поставленої задачі.

Поліноміальні алгоритми також можуть істотно розрізнятися залежно від степеня полінома, що апроксимує залежність . Розглянемо оцінювання часової складності алгоритму. Така оцінка проводиться із застосуванням відношення O («велике O »): кажуть, що $f_A(N)$ зростає як $g(N)$ для великих N і записується це як $f_A(N) = O(g(N))$. Якщо існує позитивна константа $Const > 0$, така, що $\lim_{N \rightarrow \infty} f_A(N)/g(N) = Const$, то оцінка $O(g(N))$ називається асимптотичною часовою складністю алгоритму.

Оцінка $O(g(N))$ для функції $f_A(N)$ застосовується, коли точне значення $f_A(N)$ невідоме, а відомо лише порядок зростання часу, затрачуваного на розв'язання задачі розмірністю N за допомогою алгоритму A . Точні значення $f_A(N)$ залежать від конкретної реалізації, тоді як $O(g(N))$ є характеристикою самого алгоритму. Наприклад, якщо часова асимптотична складність алгоритму дорівнює $O(N^2)$ (такий алгоритм називається квадратичним), то при

збільшенні N час розв'язання задачі збільшується як квадрат N . Факт експоненціальної складності алгоритму в термінах введеної символіки можна записати як $f_A(N) = O(k^N)$, де k — як правило ціле число більше за одиницю.

Інший вид оцінки пов'язаний з введенням «малого o »: кажуть, що $f_A(N)$ зростає не швидше від $g(N)$ для великих N , що записується $f_A(N) = o(g(N))$, якщо $\lim_{N \rightarrow \infty} f_A(N)/g(N) = 0$. Наприклад, очевидно, що $x^2 = o(x^5)$, $\sin(x) = o(x)$. Інший приклад: алгоритм A є поліноміальним, якщо $f_A(N) = o(P_k(N))$, де $P_k(N)$ — деякий поліном від N степеня k . Так, алгоритм, асимптотична складність якого дорівнює $o(N \log N)$, належить до поліноміальних.

Для оцінки складності переважної більшості реальних алгоритмів достатньо логарифмічної, степеневої та показникової функцій, а також їх сум, добутків та підстановок. Усі вони монотонно зростають і задаються простими аналітичними виразами.

Алгоритм сортування — це алгоритм, що розв'язує задачу сортування, тобто здійснює впорядкування лінійного списку (масиву) елементів.

Постановка задачі.

Вхід алгоритму: послідовність з n чисел $a_1, a_2, a_3, \dots, a_n$.

Вихід алгоритму: перестановка $(a_{\pi(1)}, a_{\pi(2)}, \dots, a_{\pi(n)})$ вхідної послідовності таким чином, що $a_{\pi(1)} \leq a_{\pi(2)} \leq \dots \leq a_{\pi(n)}$ (π — перестановка послідовності чисел $1 \dots n$).

Вхідна послідовність найчастіше представляється у вигляді n -елементного масиву, хоча може мати й інше представлення, наприклад, у вигляді зв'язного списку.

Структури даних

На практиці елементи, що впорядковуються, рідко бувають просто числами. Набагато частіше, кожен такий елемент є записом (англ. *record*). В

кожному записі є ключ (англ. *key*), по якому власне і здійснюється впорядкування, в той же час є й інша супутня інформація. Алгоритм сортування на практиці має бути реалізован так, щоб разом з ключами переміщати і супутню інформацію. Якщо кожен запис містить супутню інформацію великого об'єму, то з метою звести до мінімуму переписування великих об'ємів інформації, впорядкування відбувається не у самому масиві елементів, а в масиві вказівників на елементи.

Сам метод сортування не залежить від того, чи впорядковуються тільки числа, чи також і супутня інформація, тому при описі алгоритмів для простоти припускають, що елементи є числами.

Характеристики алгоритмів

Мабуть, ніяка інша проблема не породила такої кількості різноманітних рішень, як завдання сортування. Чи існує якийсь "універсальний", найкращий алгоритм? Взагалі кажучи, немає. Проте, маючи приблизні характеристики вхідних даних, можна підібрати метод, що працює оптимальним чином.

Для того, щоб обґрунтовано зробити такий вибір, розглянемо параметри, по яких проводиться оцінка алгоритмів.

1. *Час* сортування - основний параметр, що характеризує швидкість алгоритму.

2. *Пам'ять* - лава алгоритмів вимагає виділення додаткової пам'яті під тимчасове зберігання даних. При оцінці використовуваної пам'яті не враховуватиметься місце, яке займає початковий масив і незалежні від вхідної послідовності витрати, наприклад, на зберігання коду програми.

3. *Стабільність* - стабільне сортування не міняє взаємного розташування рівних елементів. Така властивість може бути дуже корисною, якщо вони складаються з декількох полів, як на мал. 1, а сортування відбувається по одному з них, наприклад, по *x*.



Взаємне розташування рівних елементів з ключем 1 і додатковими полями "a", "b", "c" залишилося тим самим: елемент з полем "a", потім - з "b", потім - з "c".



Взаємне розташування рівних елементів з ключем 1 і додатковими полями "a", "b", "c" змінилося.

4. *Природність поведінки* - ефективність методу при обробці вже відсортованих, або частково відсортованих даних. Алгоритм поводить себе природно, якщо враховує цю характеристику вхідної послідовності і працює краще.

Ще однією важливою властивістю алгоритму є його сфера вживання. Тут основних позицій дві:

- внутрішні сортування працюють з даними в оперативній пам'яті з довільним доступом;
- зовнішні сортування упорядковують інформацію, розташовану на зовнішніх носіях. Це накладає деякі додаткові обмеження на алгоритм:

○ **доступ** до носія здійснюється послідовним чином: у кожен момент часу можна вважати або записати тільки елемент, наступний за поточним

○ **об'єм** даних не дозволяє їм розміститися в ОЗУ

Крім того, доступ до даних на носіїві проводиться набагато повільнішим, ніж операції з оперативною пам'яттю.

Даний клас алгоритмів ділиться на два основні підкласи:

- Внутрішні сортування оперує з масивами, що цілком поміщаються в оперативній пам'яті з довільним доступом до

будь-якого вічка. Дані зазвичай сортуються на тому ж місці, без додаткових витрат.

- Зовнішнє сортування оперує з пристроями великого об'єму, що запам'ятовують, але з до ступом не довільним, а послідовним (сортування файлів), тобто в даний момент ми 'бачимо' тільки один елемент, а витрати на перемотування в порівнянні з пам'яттю не виправдано великі. Це приводить до спеціальних методів сортування, що зазвичай використовують додатковий дисковий простір.

Для значної кількості алгоритмів середній і найгірший час впорядкування n -елементного масиву є $O(n^2)$, це пов'язано з тим, що в них передбачені перестановки елементів, що стоять поряд (різниця між індексами елементів не перевищує деякого заданого числа). Такі алгоритми зазвичай є стабільними, хоча і не ефективними для великих масивів.

Інший клас алгоритмів здійснює впорядкування за час $O(n \log n)$. В цих алгоритмах використовується можливість обміну елементів, що знаходяться на будь-якій відстані один від одного.

Теорема про найкращий час сортування

Якщо алгоритм сортування в своїй роботі спирається тільки на операції порівняння двох об'єктів (\leq) і не враховує жодної додаткової інформації про елементи, то він не може впорядкувати масив елементів швидше ніж за $O(n \log n)$ в найгіршому випадку.

Доведення

На кожному кроці алгоритм проводить одне порівняння, результатом якого є один з двох варіантів:

1. $A \leq B$
2. $A > B$

В залежності від результату порівняння алгоритм буде робити подальші дії. Значить всю роботу алгоритму можна представити у вигляді бінарного дерева в листах якого лежать можливі перестановки вхідного масиву.

Отже, дерево має $n!$ листів, значить висота дерева є $\log(n!)$. Час роботи в найгіршому випадку пропорційний висоті дерева:

$$\begin{aligned} & \frac{(n!)}{n} \\ & \frac{e}{\sqrt{2\pi n}} \\ & O(n \log n) \\ & \log n \\ & O(\log n) \end{aligned}$$

Ці швидкі алгоритми використовуються в реальних задачах. Проте більшість з них нестабільні. Стабільні алгоритми, що працюють за час $O(n \log n)$ потребують $O(n)$ додаткової пам'яті.

Відомий стабільний алгоритм сортування, що не вимагає додаткової пам'яті працює за час $O(n \log^2 n)$.

Ще один клас алгоритмів використовує в своїй роботі деяку додаткову інформацію про елементи, що впорядковуються (наприклад, те що вони є різними числами в деякому діапазоні). Завдяки цьому, вони працюють за час $O(n)$.

Відомі алгоритми сортування

За час $O(n^2)$

- сортування вибором
- сортування вставкою
- сортування обміном (бульбашки)

За час $O(n \log n)$

- сортування купою
- швидке сортування
- сортування злиттям

За час $O(n)$ з використанням додаткової інформації про елементи

- сортування підрахунком
- сортування за розрядами
- сортування комірками

За час $O(n \log^2 n)$

- сортування злиттям модифіковане
- сортування Шелла

2.2. Порівняння алгоритмів сортування

Спробуємо порівняти ефективність методів сортування. Хай n як і раніше позначає число сортованих елементів, а C і M — відповідно кількість необхідних порівнянь ключів і пересилок елементів. Для всіх трьох простих методів сортування можна дати замкнуті аналітичні формули. Вони приведені в табл. 2.1. Заголовки стовпців Min, Max, Середин. визначають відповідно мінімуми, максимуми і очікувані середні значення для всіх $n!$ перестановок n елементів.

Таблиця 2.1.

Порівняння простих методів сортування

	Min	Середин	Max
сортування вставкою	$C = n - 1$ $M = 2(n - 1)$	$\frac{(n^2 + n - 2)/4}{(n^2 - 9n - 10)/4}$	$\frac{(n^2 - n)/2 - 1}{(n^2 + 3n - 4)/2}$
сортування вибором	$C = (n^2 - n)/2$ $M = 3(n - 1)$	$\frac{(n^2 - n)/2}{n(\ln n + 0,57)}$	$\frac{(n^2 - n)/2}{n^2/4 + 3(n - 1)}$
сортування обміном (бульбашки)	$C = (n^2 - n)/2$ $M = 0$	$\frac{(n^2 - n)/2}{(n^2 - n) * 0,75}$	$\frac{(n^2 - n)/2}{(n^2 - n) * 1,5}$

Для вдосконалених методів немає досить простих і точних формул. Все, що можна сказати, — це що вартість обчислень дорівнює $c \cdot n^{1.2}$ в разі сортування Шелла і $c \cdot n \log n$ у випадках пірамідального і швидкого сортувань.

Ці формули дають лише приблизну оцінку ефективності як функції від n ; вони допускають класифікацію алгоритмів сортування на простих (n^2) і вдосконалених, або «логарифмічні» ($n \log n$). Проте для практичних цілей корисно мати деякі експериментальні дані, які можуть пролити світло на коефіцієнти c , що дозволяють проводити подальшу оцінку різних методів. Крім того, в цих формулах не враховуються витрати на інші операції, відмінні від порівнянь ключів і пересилок елементів, такі, як управління циклами і так далі. Зрозуміло, ці чинники якоюсь мірою залежать від конкретних систем,

але проте деякий приклад експериментально отриманих даних є інформативним

У таблиці 2.2. приведений час (у мілісекундах), який витратила система Паскаль на обчислювальній машині CDC 6400 на виконання сортування описаними тут методами. У трьох стовпцях вказаний час, що було потрібно для сортування вже розсортованого масиву, випадкової перестановки і масиву із зворотним ладом елементів.

Таблиця 2.2.

Час виконання програм сортування

	Впорядкований масив	Випадковий масив	Впорядкований в зворотному порядку масив
Просте включення	12 23	366 1444	704 2836
Бінарне включення	56 125	373 1327	662 2490
Простий вибір	489 1907	509 1956	695 2675
Метод бульбашки	540 2165	1026 4054	1492 5931
Метод бульбашки з обмеженням	5 8	1104 4270	1645 6542
Шейкер-сортування	5 9	961 3642	1619 6520
Сортування Шелла	58 116	127 349	157 492
Пірамідальне сортування	116 253	110 241	104 226
Швидке сортування	31 69	60 146	37 79
Сортування злиттям	99 234	102 242	99 232

Ліве число в кожній колонці дане для масиву з 256 елементів, а праве — для 512 елементів. Ці дані демонструють явну відзнаку методів P^2 від методів *nlogn*. Прим і тні наступні моменти:

1. Перевага сортування бінарними включеннями в порівнянні з сортуванням простими включеннями дійсно нікчемно, а в разі вже наявного ладу зокрема відсутній.

2. Сортування методом бульбашки є найгіршим серед всіх порівнюваних методів. Її поліпшена версія — шейкер-сортування все-таки гірше, ніж сортування простими включеннями і простим вибором.

3. Швидке сортування перевершує пірамідальне сортування відносно 2 до 3. Вона сортує масив з елементами, розташованими в зворотному ладі практично так само, як вже розсортований.

Слід додати, що ці дані були отримані при сортуванні елементів, що складаються тільки з ключа без супутньої інформації. Це — не дуже реалістичне допущення; у таблиці 2.3. показано, як впливає збільшення розміру елементів на швидкість роботи програм. У вибраному прикладі супутні дані займають в 7 разів більше пам'яті, чим ключ.

Таблиця 2.3.

Час виконання програм сортування (ключі з інформацією)

	Впроядкований масив	Випадковий масив	Впроядкований в зворотньому порядку масив
Просте включення	12 46	366 1129	704 2150
Бінарне включення	56 76	373 1105	662 2070
Простий вибір	489 547	509 607	695 1430
Метод бульбашки	540 610	1026 3212	1492 5599
Метод бульбашки з обмеженням	5 5	1104 3237	1645 5762
Шейкер-сортування	5 5	961 3071	1619 5757
Сортування Шелла	58 186	127 373	157 435
Пірамідальне сортування	116 264	110 246	104 227
Швидке	31 55	60 137	37 75

	Впроядкований масив	Випадковий масив	Впроядкований в зворотньому порядку масив
сортування			
Сортування злиттям	99 196	102 195	99 187

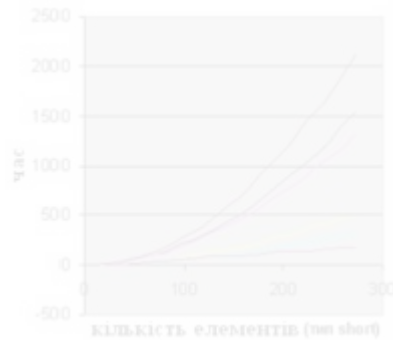
Ліве число в кожній колонці показує час, потрібний для сортування записів без супутніх даних, правий, — відображає сортування з супутніми даними; n — 256. Звернете увагу на наступні деталі:

1. Сортування простим вибором дає істотний вигрaш і виявляється кращим з простих методів.
2. Сортування методом бульбашки як і раніше є найгіршим методом (вона ще більше здала свої позиції), і лише її «удосконалення», зване шейкер-сортування, ще ледве гірше в разі масиву із зворотним порядком,
3. Швидке сортування навіть укріпило свою позицію як найшвидший метод і виявилось дійсно кращим алгоритмом сортування.

Алгоритм	Переваги	Недоліки
Сортування Вибором	Дуже простий. Швидко сортує невеликі списки	Повільно працює з великими списками
Сортування вставками	Дуже простий. Швидко сортує невеликі списки	Дуже повільно працює з великими списками
Бульбашкове сортування	Швидко працює для майже відсортованих списків	Повільно працює в решті випадків
Швидке сортування	Швидко сортує великі списки	Працює некоректно при великій кількості однакових значень
Метод Шелла	Сортує дробові числа	Вимагає про стору пам'яті для зберігання тимчасових значень
Сортування злиттям	Швидко сортує великі списки	Працює повільніше, ніж швидке сортування
Сортування підрахунком	Дуже швидко працює, якщо розкид вхідних значень не великий	Повільно працює у випадку якщо розкид складає $>\log(N)$
Сортування Шейкером	Сортує дані на жорсткому диску	Працює повільніше, ніж швидке сортування

Порівняння часу сортувань

Змальований нижче графік ілюструє різницю в ефективності вивчених алгоритмів.



- коричнева лінія: сортування бульбашкою;
- синя лінія: шейкер-сортування;
- рожева лінія: сортування вибором;
- жовта лінія: сортування вставками;
- блакитна лінія: сортування вставками із сторожовим елементом;
- фіолетова лінія: сортування Шелла.

2.3. Висновки до розділу

В данному розділі розглядається оцінка складеності алгоритмів сортування. Проводиться аналіз алгоритмів сортування.

Згідно вимог, прості алгоритми сортування (такі, як сортування вибором і сортування включенням) не є дуже ефективними.

Алгоритм сортування обмінами, хоча і завершує свою роботу (оскільки він використовує лише цикли з параметром і в тілі циклів параметри примусово не змінюються) і не використовує допоміжної пам'яті, але займає багато часу. Навіть, якщо внутрішній цикл не містить жодної перестановки, то дії будуть повторюватись до тих пір, поки не завершиться зовнішній цикл.

Алгоритм сортування вибором ефективніше сортування обмінами за критерієм $M(n)$, тобто за кількістю пересилань, але також є не дуже ефективним. З цих причин було розроблено деякі нові алгоритми сортування, що отримали назву швидких алгоритмів сортування. Це такі алгоритми, як сортування деревом, пірамідальне сортування, швидке сортування Хоара та метод цифрового сортування.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ДОДАТКУ ВІЗУАЛІЗАЦІЇ АЛГОРИТМІВ СОРТУВАННЯ

3.1. Вибір технології для побудови візуалізатора

Візуалізатор алгоритму - це програма, написана в рамках певної технології. Як зазначалося вище, при використанні візуалізаторів на лекціях, а також при заочному навчанні можливе використання будь-яких мов програмування і відповідно, технологій. Обговорення переваг і недоліків різних підходів до побудови додатків виходять за рамки даної роботи. При побудові додатків, придатних для використання в дистанційному навчанні потрібне використання інтернет-орієнтованих технологій і мов.

При виборі технологічної платформи для створення візуалізаторів сформулюємо вимоги, яким повинна задовольняти ця платформа:

1. Наявність готових бібліотек з побудови користувацького інтерфейсу.
2. Можливість вбудовувати додатки в Web-сторінки.
3. Крос-платформеність.
4. Можливість роботи без використання браузера.
5. Широке поширення і підтримка.

Існують різні технології для реалізації додатків в Інтернет. Прикладами можуть служити Adobe Flash [33], Microsoft Silverlight [34], JavaScript [35]. Безперечними достоїнствами цих технологій є їх широке розповсюдження і спрощене написання Інтернет-додатків. Їх недоліком є їх вузька сфера застосування - вони працюють тільки в рамках браузерів. Незважаючи на недавній вихід у світ технології Adobe AIR, що дозволяє запускати Flash-додатки без браузерів, сама технологія істотно поступається по гнучкості і поширеності технології Java. Технологія Java Applets, що є частиною технологічної платформи Java, надає максимальні переваги в порівнянні з іншими мовами і технологіями.

Наведемо порівняльну таблицю потенційних рішень при виборі технології побудови візуалізаторов (табл. 3.1.), В якій цифрами 1 - 5 позначені вимоги, зазначені вище.

Таблиця 3.1.

Результати порівняння платформ для розробки візуалізаторов

Платформа	1	2	3	4	5
Flash	+	+	+	-	+
Adobe AIR	+	+	+	+	-
Silverlight	+	+	-	-	-
Delphi	+	-	-	+	-
Java Applets	+	+	+	+	+
JavaScript	±	+	+	±	±

У дипломному проєкті для організації візуалізації алгоритмів сортування було використано скрипти на мові JavaScript (рис.3.1.).

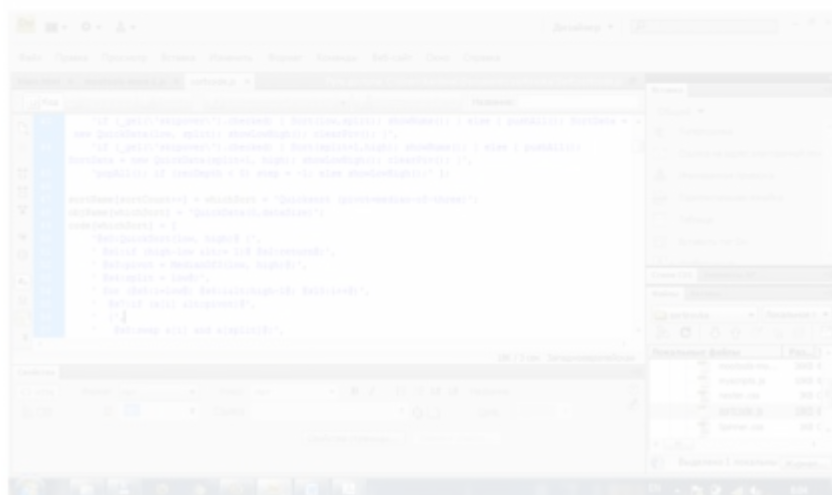


Рис. 3.1. Скрипти на мові JavaScript

Мова програмування Javascript розроблена фірмою Netscape для створення інтерактивних HTML-документів. Це об'єктно-орієнтована мова розробки вбудовуваних застосунків, що виконують як на стороні клієнта, так і на стороні сервера. Синтаксис мови дуже схожий на синтаксис мови Java – тому його часто називають java-подібним. Клієнтські застосування виконуються браузером для перегляду web- документів на машині

користувача, серверні застосування виконуються на сервері. При розробці обох типів додатків використовується загальний компонент мови, званий ядром і що включає визначення стандартних об'єктів і конструкцій (змінні, функції, основні об'єкти і засіб Liveconnect взаємодії з java-аплетами, і відповідні компоненти доповнень мови, що містять специфічні для кожного типу додатків визначення об'єктів. Клієнтські застосування безпосередньо вбудовуються в html-сторінки і інтерпретуються браузером у міру відображення частин документа в його вікні. Серверні додатки для збільшення продуктивності заздалегідь компілюються в проміжний байт-код.

Основні області використання мови Javascript при створенні інтерактивних HTML-сторінок:

- Динамічне створення документа за допомогою сценарію.
- Оперативна перевірка достовірності заповнюваних користувачем полів форм HTML до передачі їх на сервер.
- Створення динамічних html-сторінок спільно з каскадними таблицями стилів і об'єктною моделлю документа.
- Взаємодія з користувачем при вирішенні “локальних” завдань, що вирішуються додатком Javascript, вбудованому в html-сторінку.

Ідея JavaScript дуже проста. Всі операції, які можна виконувати в програмі на JavaScript, описують дії над добре відомими і зрозумілими об'єктами, якими є елементи робочої області, браузера і контейнери мови HTML. Власне об'єктна орієнтованість JavaScript на цьому і кінчається. Є лише об'єкти з набором властивостей і набір функцій над об'єктами.

Останні називаються методами. Окрім методів існують і інші функції, які більше схожі на функції з традиційних мов програмування і дозволяють працювати із стандартними математичними типами або управляти процесом виконання програми. Ще в JavaScript є події - аналог програмних переривань.

Ці події також орієнтовані на роботу в World Wide Web, наприклад, завантаження сторінки в робочу область браузера або вибір гіпертекстового посилання. Використовуючи події, автор гіпертекстової сторінки і програми

відображує її може організувати перегляд динамічних об'єктів, наприклад, рядок, що біжить, або управління багатовіконним інтерфейсом.

Всю ієрархію мови можна представити таким чином (рис 3.2): на самому верхньому рівні ієрархії знаходиться об'єкт window, що представляє вікно браузера і який є “батьком” всіх останніх об'єктів. Розташовані нижче в ієрархії об'єкти можуть мати свої підлеглі об'єкти. На малюнку показана структура об'єктів клієнта (браузера).



Рис.3.2. Ієрархія об'єктів Javascript на стороні клієнта

Окрім цих класів об'єктів користувач може створювати і свої власні. Але звичайна більшість програм використовують цю систему класів і не створюють нових.

Осібно розташований об'єкт navigator з двома дочірніми (підлеглими) об'єктами. Він відноситься до самого браузера, і його властивості дозволяють визначити характеристики програми перегляду. Кожна сторінка на додаток до об'єкту navigator обов'язково має ще чотири об'єкти:

- window — об'єкт верхнього рівня, властивості якого застосовуються до всього вікна, в якому відображується документ.
- document — властивості якого визначаються вмістом самого документа: зв'язки, колір фону, форми і так далі

- location – властивості якого пов'язані з url-адресою документа, що відображується.
- history – представляє адреси html-сторінок, що раніше завантажувалися.

Окрім вказаних об'єктів сторінка може мати додаткові об'єкти, залежні від її вмісту, які є дочірніми об'єктами об'єкту document . Якщо на сторінки розташована форма, то всі її елементи є дочірніми об'єктами цієї форми. Для завдання точного імені об'єкту використовується точкова нотація з повною вказівкою всього ланцюжка спадкоємства об'єкту. Це можливо, оскільки об'єкт верхнього рівня має властивість, значенням якого є об'єкт нижнього рівня. Посилання на об'єкт здійснюється по імені, заданому параметром NAME тега HTML.

Для створення візуалізації алгоритмів сортувань нам необхідні наступні компоненти:

- Прапорці;
- Списки;
- Картинки;
- Кнопки;
- Об'єкт textarea;
- Об'єкт text.

Прапорці

Елемент управління "прапорець" використовується у разі, коли із запропонованих варіантів можна вибрати як один, так і декілька. Кожен варіант вибору задається прапорцем, який можна або встановити, або скинути. Прапорець визначається в теге <input> значенням checkbox параметра type. Обов'язковим параметром є параметр value, значення якого буде передано на обробку в разі вибору натисненням кнопки.

Списки

Якщо елементів багато, то виставлення їх за допомогою прапорців або перемикачів збільшує розмір форми. В цьому випадку варіанти вибору можуть бути представлені у вікні браузера компактніше за допомогою тега `<select>`. Нагадаємо, що тег має декілька параметрів. Параметр `name` є обов'язковим. Для того, щоб встановити число одночасно видимих елементів, слід задати параметр `size=n`. Коли `n` дорівнює 1, то відображується спадаюче меню або список вибору; при `n > 1` виводиться список з `n` одночасно видимими значеннями. Якщо параметр `size` не заданий, то за умовчанням набуває значення рівне 1. Вказівка параметра `multiple` означає, що з меню або списку можна вибрати декілька елементів. Елементи меню задаються у середині тега `<select>` за допомогою тега `<option>`. Загальний вигляд тега такий:

```
<option selected value=строка>
```

Параметр `selected` означає, що даний елемент списку вважається вибраним за умовчанням. Параметр `value` містить значення, яке передається, якщо даний елемент вибраний із списку або меню.

Картинки

Кнопки-картинки — це ті ж кнопки, але лише з можливістю відправки даних на сервер. Власне, такі кнопки в Javascript складають два різновиди контейнера `INPUT`: `image` і `submit`. У Javascript об'єкт, пов'язаний з даними кнопками, називається `Submit`.

```
<FORM>
```

Активна кнопка:

```
<INPUT Type=image Src=images.gif onclick="return false;">
</form>
```

Як ми вже відзначали, даний об'єкт володіє тими ж властивостями, методами і подіями, що і об'єкт `Button`. Але реакція в різних браузерах при обробці подій може бути різною. Так, в події `onclick` в Internet Explorer можна відмінити передачу даних на сервер, видавши як значення повернення `false`.

Netscape Navigator на таку поведінку обробника події взагалі не реагує і відмінити передачу можна лише в атрибуті onsubmit контейнера FORM:

```
<FORM onsubmit="return false">
```

Активна кнопка:

```
<INPUT Type=image Src=images.gif border=0>
```

```
</form>
```

Кнопки

Використання кнопок в Web взагалі немислимо без вживання Javascript .

Кнопка вводиться у форму головним чином для того, щоб можна було обробити подію click:

```
<FORM>
```

```
<INPUT          Type=button          Value="Окно          запобігання"
onclick="window.alert("Открилі окно");">
```

```
</form>
```

Текст, що відображується на кнопці, визначається атрибутом VALUE контейнера INPUT. З цим атрибутом пов'язана властивість value об'єкту Button. **Согласно** специфікації, змінювати значення даного атрибуту не можна.

Об'єкт text

Об'єкт text - це поле введення, визначуване в теге <Input type="text"> і надаючи користувачеві можливість вводити текстові дані. Об'єкт text є властивістю об'єкту form і повинен розміщуватися в контейнері <form> . . . </form>. Об'єкти text містять дані, які можна і читати, і динамічно змінювати в Js-програмах.

```
<input [type="text"]
```

```
name="textName"
```

```
value="textValue"
```

```
size=integer
```

```
[onBlur="handlerText"]
```

```
[onChange="handlerText"]
```

```
[onFocus="handlerText"]  
[onSelect="handlerText"]>
```

Об'єкт textarea

Об'єкт textarea відповідає області тексту, визначеній у формі. Об'єкти textarea є властивостями об'єкту form і мають бути поміщені в контейнер `<form> . . . </form>`. Елементи цього типу використовуються для введення декількох рядків тексту у вільному форматі. Також його часто використовують для виведення прикладів тексту наприклад js-програмі, сформірованого тексту пропонуваного для розміщення наприклад банера і ін. Для звернення до методів і властивостей об'єкту textarea застосовуються типові для елементів форми вирази:

- `textareaName.propertyName`
- `textareaName.methodName(parameters)`
- `formName.elements[i].propertyName`
- `formName.elements[i].methodName(parameters)`

де `textareaName` - це значення атрибуту `name` тега `<textarea>`, а `formName` - ім'я форми, в котрій визначений об'єкт textarea. Вміст об'єктів textarea в js-програмах може динамічно змінюватися шляхом надання нового значення їх властивості `value`.

3.2. Вибір середовища розробки візуалізатора алгоритмів

Провівши огляд програмних засобів, призначених для редагування і верстки HTML кодів, було з'ясовано, що кожен редактор має як свої переваги, так і недоліки. Зваживши всі за і проти, мій вибір зупинився на HTML редакторі Dreamweaver MX.

Dreamweaver – це сучасне програмне середовище для створення веб-сайтів. Інші програми дуже сильно уступають Dreamweaver по зручності, наочності і простоті в освоєнні, тому що, фірма-розробник Macromedia, постійно займається його удосконалюванням.

Система Dreamweaver — це візуальний редактор гіпертекстових документів, в яку інтегровано кілька програмних засобів і модулів, що

забезпечують весь операційний цикл розробки і підтримки віртуальних проектів.

Могутня професійна програма Dreamweaver володіє всіма необхідними засобами для генерації сторінок HTML будь-якої складності і масштабу.

Вона забезпечує режим візуального проектування (WYSIWYG, ця громіздка аббревіатура утворена по перших буквах англійської фрази What you see is what you get (що бачите, те й отримуєте), відрізняється дуже чистою роботою з вихідним текстом Web-документів, має убудовані засоби підтримки великих мережних проектів.

Це значить, що зображення сторінки HTML у вікні документа не сильно відрізняється від її представлення в найбільш популярних програмах перегляду — броузерах Microsoft Internet Explorer і Netscape Navigator.

У програмі послідовно підтримується візуальне проектування. Візуальним прийнято називати такий спосіб створення гіпертекстових документів, у якому робота з текстом і образами об'єктів переважає над безпосереднім кодуванням. В ідеалі, користувач повинний бути цілком вільний від необхідності звертання до кодів HTML, а проектування зобов'язане замінити програмування. Пряма робота з кодами не виключена в цілому, але зведена до розумного мінімуму.

Програма не тільки має могутній арсенал засобів візуального проектування, але і здатна відображати Web-сторінки майже як спеціалізовані програми перегляду: Microsoft Internet Explorer чи Netscape Navigator, або Opera, та інші.

В оболонку Dreamweaver інтегрований повнофункціональний редактор HTML, що володіє всіма необхідними інструментами для роботи з дескрипторами гіпертекстової розмітки.

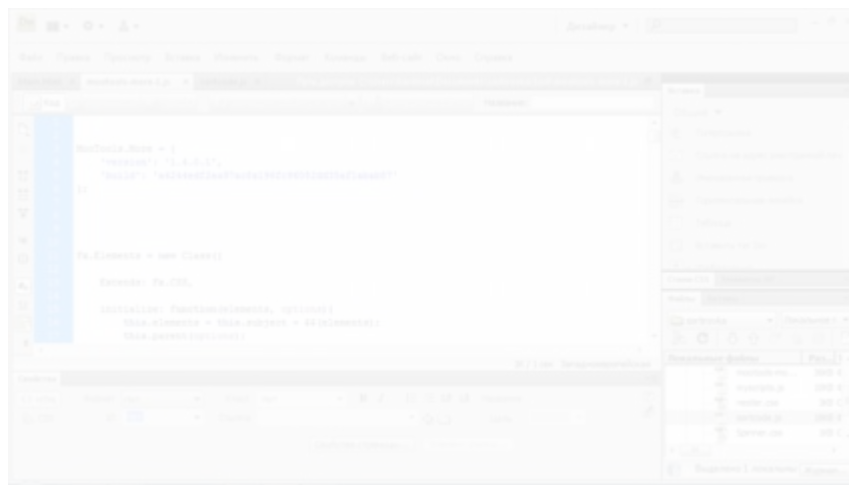


Рис. 3.3. HTML редактор Dreamweaver MX

Програма Dreamweaver заснована на принципах відкритої архітектури. Це означає, що цілком відкритий інтерфейс прикладного програміста (Application Programming Interface, API), за допомогою якого сторонні програмісти можуть вносити радикальні зміни до функціональних можливостей програми і її інтерфейсу: додати новий інструмент, створити палітру чи розділ меню, запрограмувати новий об'єкт чи мультимедійний ролик та інше. Але процес розробки розвинутого мережного проекту, як правило, не закінчується створенням усіх складових гіпертекстових документів та структури.

В оболонку Dreamweaver інтегрований спеціальний засіб, призначений для обслуговування розповсюдження у мережі документів і сайтів. Він має всі необхідні інструменти для дистанційного відновлення версій документів. Цей засіб підтримує розподілену роботу декількох виконавців над одним мережним проектом, має механізм синхронізації версій документів і захисту від несанкціонованого доступу.

3.3. Каскадні таблиці стилів CSS

Каскадні таблиці стилів або про сто *таблиці стилів* (CSS — Cascading Style Sheets) — це набір правил, що описують форматування різних

фрагментів HTML-коду й зберігаються окремо від його. Одне таке правило, що відображає форматування якогось одного фрагмента або однотипної групи фрагментів коду, називається **СТИЛЕМ**.

Таблиці стилів описуються особливою мовою CSS і зберігаються в особливих файлах з розширенням **CSS**, хоча можуть бути впроваджені в саму Web-сторінку, таблицю стилів, збережену в окремому файлі, можна використати в багатьох Web-сторінках. Більше того, вона може перебувати взагалі на іншому сайті.

Web-сторінка може посилатися одночасно на кілька таблиць стилів. Наприклад:

```
<LINK REL="stylesheet" HREF="styles1.css">
```

```
<LINK REL="stylesheet" HREF="styles2.css">
```

за допомогою таблиці стилів можна змінити зовнішній вигляд будь-якого тегу HTML. Для цього потрібно просто перевизначити його в таблиці стилів у такий спосіб:

```
H1 { color: #FF0000;
```

```
font-size: smaller }
```

За допомогою таблиць стилів можна формувати не тільки текст. Будь-якому елементу сторінки - зображенню, таблиці, горизонтальній лінії - може бути привласнений стильовий клас.

Три способи завдання стилю

Усього стандарт CSS визначає три способи завдання стилю для елемента сторінки.

- Зовнішня (або прив'язана) таблиця стилів. Стилі зберігаються в окремому файлі з розширенням css і прив'язуються до Web-сторінки за допомогою особливого тегу <LINK>.

- Внутрішня (або впроваджена) таблиця стилів. Таблиця стилів у цьому випадку має той же самий формат, що й зовнішня, але розташовується в секції заголовка тієї ж Web-сторінки й міститься усередину спеціального тегу <STYLE>.

- Внутрішні (також убудовані або впроваджені) стилі. Визначення стилю міститься прямо в потрібний тег, для чого використовується спеціальний атрибут STYLE.

Вся робота зі стилями виконується в панелі CSS Styles Dreamweaver: пункт меню Window, пункт CSS Styles.

Щоб створити новий стиль, виберіть пункт New CSS Style контекстного або додаткового меню.

Недоліки CSS

Нажаль не всі веб-броузери коректно підтримують каскадні таблиці стилів, тому що це новий стандарт (1997 р.). У деяких випадках необхідно переконвертувати сторінку у вікні броузера.

У диплому проекті були використані каскадні таблиці стилів для оформлення сторінки візуалізації алгоритмів.

3.4. Опис інтерфейсу Web-модуля навчальної системи для візуалізації алгоритмів сортування

Дуже важливу роль в ефективності роботи програми грає правильно розроблений інтерфейс. Саме від цього буде залежати виконання декількох з основних вимог - зручність і простота в освоєнні. Складний, перевантажений інтерфейс може зменшити ефективність роботи. Головне правило, від якого варто відштовхуватися - інтерфейс не повинен заважати роботі користувача й не повинен відволікати його увагу від роботи, одночасно не втрачаючи при цьому у функціональності.

Виходячи із цього, було вирішено використовувати типові для windows-програм візуальні компоненти. Це дасть можливість більшості користувачів швидко розібратися й включитися в роботу. Інтерфейс прямо залежить від функцій, що виконуються програмою.

Список функцій програми.

Можливість сортування числових та символьних масивів наступними методами:

1. Швидке сортування (Quicksort (pivot=last));

2. Швидке сортування (Quicksort (pivot=median-of-three));
3. Метод бульбашки (Bubble Sort);
4. Метод бульбашки (Bubble Sort (smart));
5. Гном сортування (Gnome Sort);
6. Сортування Шейкером (Shaker Sort);
7. Комбінований (CombSort11);
8. Сортування вставками (Insertion Sort);
9. Сортування Шелла (Shell Sort);
10. Бінарною купою (Heap Sort);
11. Сортування злиттям (Merge Sort);
12. Сортування за розрядами (Radix Sort).

Основу інтерфейсу повинні становити компоненти представляючі користувачеві саму необхідну інформацію й доступ до найважливіших функцій системи.

Впливаючи із цього, головне вікно програми повинне містити наступні основні компоненти:

1. Список алгоритмів сортування для вибору;
2. Методи сортування;
3. Розмірність масиву;
4. Затримка в ms;
5. Покрокове виконання;
6. Автозапуск;
7. Перезапуск;
8. Колекція даних;
9. Відображення алгоритму сортування;
10. Відображення даних сортування.

Схема інтерфейсу програми-клієнта показана на рис. 3.4.



Рис. 3.4. Загальна концепція інтерфейсу візуалізації а лгоритмів сортування

Слід зазначити, що вся концепція інтерфейсу виконана в простому стилі й у цілому типова для програм такого роду.

3.5. Методичні вказівки по використанню Web-модуля навчальної системи для візуалізації алгоритмів сортування

Після запуску Main.html з'являється сторінка на якій є можливість проводити сортування масивів даних та відображати їх демонстрацію.

Користувач має можливість обрати алгоритми, які необхідно виконати у середовищі демонстрації, та обрати для них відповідні дані (рис. 3.5.-3.6).



Рис. 3.5. Вибір алгоритму сортування

Колекція даних, з якої користувач може обрати масиви, необхідна для оцінки ефективності різних алгоритмів на однакових даних, та формулювання висновків про ефективність виконання одного і того ж

алгоритму на різних даних. Завантажений алгоритм відображається у правій частині вікна.



Рис. 3.6. Вибір даних які підлягають сортуванню

Колекція даних може бути числовою, символьною або гістограма (рис. 3.6.-3.8).



Рис. 3.7. Вибір символьного масиву даних



Рис. 3.8. Вибір гістограмного масиву даних

Наступний крок – заповнення контейнерів даними. Для цього необхідно в розкриваю чому списку «Розміщення даних» (рис. 3.9), що формує дані для візуалізації виконання алгоритму у середовищі демонстрації одним із способів:

- за зростанням;
- за спаданням;
- випадковим чином (з перемішуванням);
- випадковим чином (без перемішування);
- константами.



Рис. 3.9. Формування даних для візуалізації виконання алгоритму

Також треба установити розмір масиву (максимум 20).

Користувач може встановити «Неперервний або покроковий тип роботи». При виборі «Неперервний тип роботи» за допомогою кнопки «Автозапуск» можна виконувати візуалізацію алгоритму автоматично.

Якщо обрати перемикач «Пропустити вивод функцій», то будуть пропускатися функції всередині алгоритму.

При виборі «Покрокового типу роботи» за допомогою кнопки «Покрокове виконання» можна виконувати візуалізацію кожного окремого кроку алгоритму рис. 3.10.



Рис. 3.10. Покрокове виконання алгоритму

3.6. Висновки до розділу

У даному розділі сформульовані функціональні вимоги до розроблюваної програми, серед яких основними є наступні:

1. Можливість сортування числових та символьних масивів наступними методами:
 - Швидке сортування (Quicksort (pivot=last));
 - Швидке сортування (Quicksort (pivot=median-of-three));
 - Метод бульбашки (Bubble Sort);
 - Метод бульбашки (Bubble Sort (smart));
 - Гном сортування (Gnome Sort);
 - Сортування Шейкером (Shaker Sort);
 - Комбінований (CombSort11);
 - Сортування вставками (Insertion Sort);
 - Сортування Шелла (Shell Sort);
 - Бінарною купою (Heap Sort);
 - Сортування злиттям (Merge Sort);
 - Сортування за розрядами (Radix Sort).
2. Можливість демонструвати наочний матеріал;
3. Легкість в освоєнні й роботі з програмою.

Був складений загальний алгоритм роботи програми та розроблен інтерфейс, на основі передбачуваної функціональності був обраний програмний засіб для реалізації поставленої мети в роботі.

Мовою програмування було обрано JavaScript і спеціалізоване середовище Dreamweaver MX.

ВИСНОВКИ

У магістерській роботі було розглянуто проектування навчальної системи візуалізації роботи алгоритмів сортування.

Простота використання важлива при самостійному навчанні, так як в цьому випадку користувачі дуже рідко чит ають інструкції по застосуванню. Можливості відображення шагу а лгоритму і коментування шагу програми є дуже важливими. Перша - для пояснення простих алгоритмів і введення в інформатику, а друга - для розуміння складних алгоритмів. Доступність, платформонезалежних і незалежність від мережі дозволяє використовувати візуалізатор як на лекціях і практичних заняттях, так і для самостійного навчання.

Використання візуалізаторів при формуванні алгоритмічних компетенцій майбутніх програмістів дозволяє розвивати пізнавальні можливості студентів, вміння самостійно аналізувати та інтерпретувати результати, спонукає до до слі дницької діяльності. Таким чином, у студентів формуються якісно нові професійно значимі вміння та навички, реалізується підготовка майбутнього спеціаліста-програміста для успішної професійної діяльності.

Для досягнення поставленої мети були вирішені наступні завдання:

- Розглянуто застосування візуалізаторів в навчальному процесі.
- Проведено аналіз та порівняння алгоритмів сортування.
- Визначено зміст, структура, функції і дидактичні можливо сті додатку для реалізації візуалізації а лгоритмів сортування.
- На основі проведеного аналізу інструментальних засобів та теоретичного матеріалу по алгоримам сортування інформації

розроблено і реалізовано діючий Web-модуль навчальної системи для візуалізації алгоритмів сортування, який візуалізує поширені методи сортування. Як програмне середовище було обрано мову програмування JavaScript і візуальне середовище Dreamweaver MX. До програми були розроблені методичні рекомендації з використання.

Практичну цінність своєї роботи бачу в тім, що мною був отриманий багатий досвід в реалізації алгоритмів сортування та удосконалені знання по роботі з JavaScript в спеціалізованому середовищі Dreamweaver MX, в якому розроблено Web-модуль навчальної системи для візуалізації алгоритмів сортування, який можливо використовувати в навчальному процесі.

Internet sources	36
------------------	----

1	http://ite.kspu.edu/index.php/ite/article/download/434/448	18 Sources	24.8%
2	https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B3%D0%BE%D0%B2%D0%BED1%80%D0%B5%D0%BD%D0%BD%	4 Sources	4.23%
3	https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1...		2.37%
4	http://ni.biz.ua/9/9_8/9_8348_algoritmi-sortirovki.html	2 Sources	2.14%
5	https://ukrbukva.net/page,2,89565-Sozдание-interaktivnyh-Web-prilozheniiy-s-ispol-zovaniem-yazyka-programmirovani	3 Sources	1.31%
6	https://referat.co/kurosovaya-rabota-teoriya/388177-informatika-vt-telekommunikatsii-programa-dlya-sortuvannya-dan	3 Sources	1.08%
7	https://kras-meriass.in.ua/3822.html		0.45%
8	https://studopedia.su/5_16452_obchislennyya-sumi-elementiv-masivu.html		0.32%
9	https://studall.org/all3-24388.html		0.24%
10	https://repositorio.unsaac.edu.pe/bitstream/handle/20.500.12918/5335/253T20200160_TC.pdf?isAllowed=y&sequence=1		0.09%
12	http://hdl.handle.net/2099.1/11962		0.07%

Library sources	5
-----------------	---

11 **Student submission** File ID: **1013005473** Institution: **Luhansk Taras Shevchenko National University** **5 Sources** **0.09%**

Quotes

Quotes 11

- 1 University of Joensuu (Jeliot) У прикладах до системи візуалізації Jeliot [29] наведені тільки візуалізатори бульбашкової та швидкої сортувань.
- 2 «Для того, щоб все сприймалося легше, потрібно, наскільки лише це можливо, залучати до сприймання зовнішні чуття»
- 3 «лише простим продовженням того, що було закладено природою в якості інстинкту і навіть у тих же природних проявах»
- 4 «Чим більшою кількістю чуттів ти пізнаєш суть явища чи будь-якого предмета, тим правильнішими будуть твої знання про нього»
- 5 «наочність не ізолює сприйняття і представлення від цілісної аналітико-синтетичної розумової діяльності»
- 6 «Принцип наочності передбачає навчання на основі живого сприймання конкретних предметів і явищ дійсності або їх зображень»
- 7 «Технічні засоби навчання розширюють змістовний бік наочності навчання, дозволяють передавати інформацію в більш активній формі сприйняття, вони накладають свій відбиток на мислену діяльність студентів, їх емоційний стан, змінюють їх психічне навантаження»
- 8 «безпосередня наочність, що заснована на спостереженнях дійсності та опосередкована наочність, яка визначає явище, подію, предмет вивчення у певній наглядній формі, яка відображує його сутність, зв'язки і відношення»
- 9 «наочність бере участь у формуванні первинної нейронної моделі образу, поняття, явища на етапі сприйняття. Тому наочність повинна якомога більше підкріплювати цей етап переходу інформації із зовнішнього середовища у пам'ять інтелекту»
- 10 «Візуальне мислення – це людська діяльність, продуктом якої є породження нових образів, створення нових візуальних форм, що несуть певне смислове навантаження і що роблять значення видимим»
- 11 «Відеоінтерпретатор алгоритмів пошуку та сортування»